FsNet: Feature Selection Network on High-dimensional Biological Data*

Dinesh Singh[1], **Héctor Climente-González**[1], **Mathis Petrovich**[2],
Eiryo Kawakami[3,4], **Makoto Yamada**[1,5]
[1]RIKEN AIP, [2]École des Ponts ParisTech,
[3]RIKEN Medical Sciences Innovation Hub Program, [4]Chiba University
[5]Kyoto University

1 Introduction

The recent advancements in measuring devices for life sciences have resulted in the generation of large biological datasets, which are extremely important for many medical and biological applications, including disease diagnosis, biomarker discovery, drug development, and forensics (Li & Chen, 2014). Generally, such datasets are substantially high-dimensional (i.e., many features with small number of samples) and contain complex nonlinear patterns. Machine learning methods, including genome-wide association studies ($d > 10^5$, $n < 10^4$) and gene selection ($d > 10^4$, $n < 10^3$) (Marx, 2013), have been successfully applied to discover the complex patterns hidden in high-dimensional biological and medical data. However, most nonlinear models in particular deep neural networks (DNN) are difficult to train under these conditions because of the significantly high number of parameters. Hence, the following questions naturally arise: 1) are all the features necessary for building effective prediction models? and 2) what modifications are required in the existing machine-learning methods to efficiently process such high-dimensional data?

The answer to the first question is to select the most relevant features, thereby requiring an appropriate feature selection method (Ye et al., 2019; Ming & Ding, 2019; Liao et al., 2019). This problem, called feature selection, consists of identifying a smaller subset (i.e., smaller than the original dataset) that contains relevant features such that the subset retains the predictive capability of the data/model while eliminating the redundant or irrelevant features (Yamada et al., 2014, 2018a; Climente-González et al., 2019). Most state-of-the-art feature selection methods are based on either sparse-learning methods, including Lasso (Tibshirani, 1996), or kernel methods (Masaeli et al., 2010; Yamada et al., 2018b, 2014). These *shallow* approaches satisfactorily work in practice for biological data. However, sparse-learning models including Lasso are in general linear and hence cannot capture high-dimensional biological data. Kernel-based methods can handle the nonlinearlity, but it heavily depends on the choice of the kernel function. Thus, more flexibile approaches that can train an arbitrary nonlinear transformation of features are desired.

An approach to learning such a nonlinear transformation could be based on deep autoencoders (Vincent et al., 2010). However, deep autoencoders are useful for computer-vision and natural language processing tasks, wherein a large number of training samples are available. In contrast, for high-dimensional biological data, the curse of dimensionality prevents us from training such deep models without overfitting. Moreover, these models focus on building useful features rather than selecting features from data. The training of autoencoders for feature selection results in the discrete combinatorial optimization problem, which is difficult to train in an end-to-end manner.

To train neural networks on high-dimensional data without resulting in overfitting, several approaches were proposed. Widely used ones are based on random projection and its variants (Dahl et al., 2013; Wójcik & Kurdziel, 2019). However, their performances significantly depend on the random

*Correspondence to: Dinesh Singh and Makoto Yamada {dinesh.singh, makoto.yamada}@riken.jp

Preprint. Under review.

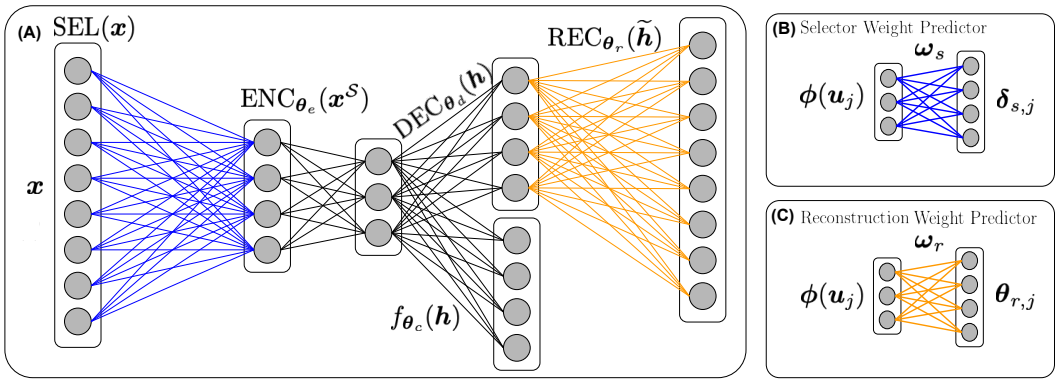

Figure 1: (A) Architecture of FsNet. (B) and (C) are the weight-predictor networks for the selection and reconstruction layers, respectively.

projection matrix, and their usability is limited to dimensionality reduction only. Therefore, they cannot be applied for feature selection. Another deep learning-based approach employs a concrete autoencoder (CAE) (Balin et al., 2019), which uses concrete random variables (Maddison et al., 2017) to select features without supervision. Although CAE is an unsupervised model, it can be extended to incorporate a supervised-learning setup. However, we observed that this simple extension is not efficient because it needs the large number of parameters in the first layer of CAE.

To address these issues, we propose a non-linear feature selection network, called FsNet, for high-dimensional biological data. FsNet comprises a selection layer that uses concrete random variables (Maddison et al., 2017), which are the continuous variants of a one-hot vector, and a reconstruction layer that stabilizes the training process. The concrete random variable allows the conversion of the discrete optimization problem into a continuous one, enabling the backpropagation of gradients using the reparameterization trick. During the training period, FsNet selects a few features using its selection layer while maximizing the classification accuracy and minimizing the reconstruction error. However, owing to the large number of parameters in the selection and reconstruction layers, overfitting can easily occur under a limited number of samples. Therefore, to avoid overfitting, we propose using two tiny networks to predict the large, virtual weight matrices of the selection and reconstruction layers. Consequently, the size of the model is significantly reduced and the network can scale high-dimensional datasets on a resource-limited device/machine. Through experiments on the real-world metagenome dataset (Lloyd-Price et al., 2019), we show that the proposed FsNet outperforms SVM, CAE and Diet-networks in performance and gives slightly lower performance than the state-of-the-art kernel methods. Moreover, we show that the required memory size of FsNet is significantly smaller than that of CAE.

2 Problem Formulation

Let $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)^\top = (\boldsymbol{u}_1, \cdots, \boldsymbol{u}_d) \in \mathbb{R}^{n \times d}$ be the given data matrix, where $\boldsymbol{x} \in \mathbb{R}^d$ represents the sample vector with d number of features and $\boldsymbol{u} \in \mathbb{R}^n$ the feature vector with n number of samples. Let $\boldsymbol{y} = (y_1, \cdots, y_n)^\top \in \mathbb{R}^n$ be the target vector such that $y_i \in \mathcal{Y}$ represents the output for $\boldsymbol{x}_i$, where $\mathcal{Y}$ denotes the domain of the output vector $\boldsymbol{y}$, which is continuous for regression problems and categorical for classification problems. In this paper, we assume that the number of samples is significantly fewer than that of the dimensions (i.e., $n \ll d$).

The final goal of this paper is to train a neural-network classifier $f(\cdot) : \mathbb{R}^d \to \mathcal{Y}$, which simultaneously identifies a subset $\mathcal{S} \subseteq \mathcal{F} = \{1, 2 \cdots d\}$ of features of a specified size $|\mathcal{S}| = K \ll d$, where the subset can reproduce the remaining $\mathcal{F} \backslash \mathcal{S}$ features with minimal loss.

3 Proposed Method: FsNet

We aim to build an end-to-end, trainable, compact, feature selection model. Hence, we employ a concrete random variable (Maddison et al., 2017) to select features, and we also use the weight-predictor models used in Diet Networks to reduce the model size (Romero et al., 2017). We build

FsNet, a simple but effective model (see Figure 1). As shown in Figure 1(A), although the selection and reconstruction layers have many connections, they are virtual layers whose weights are predicted from significantly small networks, as shown in Figures 1(B) & (C), respectively. The weight-predictor networks (B) and (C) are trained on the feature embeddings.

The optimization problem of FsNet is given by

$$\min_{\boldsymbol{\Theta}} \sum_{i=1}^{n} \text{Loss}(y_i, f_{\boldsymbol{\theta}_c}(\text{ENC}_{\boldsymbol{\theta}_e}(\boldsymbol{x}_i^{\mathcal{S}}))) + \lambda \sum_{i=1}^{n} \|\boldsymbol{x}_i - \text{REC}_{\boldsymbol{\theta}_r}(\text{DEC}_{\boldsymbol{\theta}_d}(\text{ENC}_{\boldsymbol{\theta}_e}(\boldsymbol{x}_i^{\mathcal{S}})))\|_2^2, \quad (1)$$

where $\text{Loss}(y, f_{\boldsymbol{\theta}_c})$ denotes the categorical cross-entropy loss (between y and $f_{\boldsymbol{\theta}_c}$), $\|\cdot\|_2$ the ℓ_2 norm, $\lambda \geq 0$ the regularization parameter for the reconstruction loss, $\boldsymbol{\Theta}$ all the parameters in the model, $\text{SEL}(\cdot)$ the selection layer, $\boldsymbol{x}_i^{\mathcal{S}} = \text{SEL}(\boldsymbol{x}_i)$, $\text{ENC}(\cdot)$ the encoder network, $\text{DEC}(\cdot)$ the decoder network, and $\text{REC}(\cdot)$ the reconstruction layer.

Selection Layer: We first describe the selection layer, which is used to select important features in an end-to-end manner. The feature selection problem is generally a combinatorial problem, but it is difficult to train in an end-to-end manner because it breaks the propagation of the gradients. To overcome this obstacle, a concrete random variable (Maddison et al., 2017), which is a continuous relaxation of a discrete one-hot vector, can be used for the training, as it computes the gradients using the reparameterization trick. Specifically, selecting the k-th feature of the input $\boldsymbol{x}$ can be expressed as $\boldsymbol{x}^{(k)} = \boldsymbol{e}_k^\top \boldsymbol{x}$, where $\boldsymbol{e}_k \in \mathbb{R}^d$ denotes the one-hot vector whose k-th feature is 1 and 0 otherwise. The concrete variables for the k^{th} neuron in the selection layer are defined as follows:

$$\boldsymbol{\mu}^{(k)} = \frac{\exp\left((\log \boldsymbol{\delta}_s^{(k)} + \boldsymbol{g})/\tau\right)}{\sum_{j=1}^{d} \exp\left((\log \delta_{sj}^{(k)} + g_j)/\tau\right)}, k = 1, 2, \ldots, K, \quad (2)$$

where $\boldsymbol{g} \in \mathbb{R}^d$ is drawn from the Gumbel distribution. Additionally, τ denotes the temperature that controls the extent of the relaxation, K the number of selected features, and $\boldsymbol{\Delta}_s = (\boldsymbol{\delta}_{s,1}, \ldots, \boldsymbol{\delta}_{s,d}) = (\boldsymbol{\delta}_s^{(1)}, \ldots, \boldsymbol{\delta}_s^{(K)})^\top \in \mathbb{R}^{K \times d}$, $\boldsymbol{\delta}_s^{(k)} \in \mathbb{R}_{>0}^K$ is the model parameter for concrete variables. Notably, $\boldsymbol{\mu}^{(k)}$ becomes a one-hot vector when $\tau \to 0$.

Using the concrete variables $\boldsymbol{M} = (\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \ldots, \boldsymbol{\mu}^{(K)})^\top$, the feature selection process can be simply written by using matrix multiplications as follows:

$$\text{SEL}(\boldsymbol{x}) = \boldsymbol{M}\boldsymbol{x}.$$

Because the feature selection process can be written by using matrix multiplications, it can be trained in an end-to-end manner. However, the number of parameters in the selection layer is $O(dK)$; it depends on the size of the input layer d and the number of neurons in the selection layer K. Thus, for high-dimensional data, the number of model parameters can be high, resulting in overfitting under a limited number of samples n. We address both the issues by using a tiny weight-predictor network $\boldsymbol{\varphi}_{\boldsymbol{\omega}_s}(\cdot) : \mathbb{R}^b \to \mathbb{R}_{>0}^K$ to predict the weights $\boldsymbol{\delta}_{s,j} = \boldsymbol{\varphi}_{\boldsymbol{\omega}_s}(\boldsymbol{\phi}(\boldsymbol{u}_j))$ (see Figure 1(B)), where $\boldsymbol{\phi}(\boldsymbol{u}_j) \in \mathbb{R}^b$ is the embedding representation of feature j and $b \leq n$ the size of the embedding representation. The tiny weight-predictor network can also be trained in an end-to-end manner.

Encoder Network: The goal of the encoder network $\text{ENC}_{\boldsymbol{\theta}_e}(\cdot) : \mathbb{R}^K \to \mathbb{R}^h$ is to obtain a low-dimensional hidden representation $\boldsymbol{h} \in \mathbb{R}^h$ from the output of the selection layer $\boldsymbol{x}^{\mathcal{S}}$. The encoder network is expressed as follows:

$$\text{ENC}_{\boldsymbol{\theta}_e}(\boldsymbol{x}^{\mathcal{S}}) = \sigma(\boldsymbol{W}_{L_e}^{(e)} \sigma(\cdots \boldsymbol{W}_2^{(e)} \sigma(\boldsymbol{W}_1^{(e)} \boldsymbol{x}^{\mathcal{S}}) \cdots), \quad (3)$$

where $\boldsymbol{x}^{\mathcal{S}} = \text{SEL}(\boldsymbol{x})$ denotes the output of the selection layer, $\boldsymbol{\theta}_e = \{\boldsymbol{W}_\ell^{(e)}\}_{\ell=1}^{L_e}$ the weight matrix, L_e the number of layers in the encoder network, and $\sigma(\cdot)$ an activation function.

Classifier Network: The classifier network $f_{\boldsymbol{\theta}_c}(\cdot) : \mathbb{R}^h \to \mathcal{Y}$ predicts the final output from the hidden representation $\boldsymbol{h} = \text{ENC}_{\boldsymbol{\theta}_e}(\boldsymbol{x}^{\mathcal{S}})$ as follows:

$$f_{\boldsymbol{\theta}_c}(\boldsymbol{h}) = \text{softmax}(\boldsymbol{W}_{L_y}^{(y)} \sigma(\cdots \boldsymbol{W}_2^{(y)} \sigma(\boldsymbol{W}_1^{(y)} \boldsymbol{h}) \cdots), \quad (4)$$

where $\boldsymbol{\theta}_c = \{\boldsymbol{W}_\ell^{(y)}\}_{\ell=1}^{L_y}$, and L_y denotes the number of layers in the classifier network.

Decoder Network: Generally, a decoder function is employed to reconstruct the original output. However, in this paper, the decoder function $\text{DEC}_{\boldsymbol{\theta}_d}(\cdot) : \mathbb{R}^h \to \mathbb{R}^{h'}$ computes another hidden representation $\widetilde{\boldsymbol{h}} \in \mathbb{R}^{h'}$ and defines the last reconstruction layer separately. The decoder function is defined as follows:

$$\text{DEC}_{\boldsymbol{\theta}_d}(\boldsymbol{h}) = \sigma(\boldsymbol{W}_{L_d}^{(d)}\sigma(\cdots \boldsymbol{W}_2^{(d)}\sigma(\boldsymbol{W}_1^{(d)}\boldsymbol{h})\cdots). \tag{5}$$

where $\boldsymbol{h} = \text{ENC}_{\boldsymbol{\theta}_e}(\boldsymbol{x}^S)$, $\boldsymbol{\theta}_d = \{\boldsymbol{W}_\ell^{(d)}\}_{\ell=1}^{L_d}$, and L_d denotes the number of layers in the decoder network.

Reconstruction Layer: To reconstruct the original high-dimensional feature $\boldsymbol{x}$, it must have $O(dh')$ parameters and depend on the dimension d. Thus, in a manner similar to the selection layer, we use a tiny network to predict the model parameters. The reconstruction layer is expressed as follows:

$$\text{REC}_{\boldsymbol{\theta}_r}(\widetilde{\boldsymbol{h}}) = \boldsymbol{W}^{(r)}\widetilde{\boldsymbol{h}}, \tag{6}$$

where $\widetilde{\boldsymbol{h}} = \text{DEC}_{\boldsymbol{\theta}_d}(\boldsymbol{h})$, $\boldsymbol{\theta}_r = \boldsymbol{W}^{(r)} \in \mathbb{R}^{d\times h'}$, and $[\boldsymbol{W}^{(r)\top}]_j = \boldsymbol{\varphi}_{\boldsymbol{\omega}_r}(\boldsymbol{\phi}(\boldsymbol{u}_j))$ denotes the virtual weights of the j^{th} row in the reconstruction layer. The tiny network $\boldsymbol{\varphi}_{\boldsymbol{\omega}_r}(\cdot) : \mathbb{R}^b \to \mathbb{R}^{h'}$ is trained on $\boldsymbol{\phi}(\boldsymbol{u}_j) \in \mathbb{R}^b$ to predict the weights that connect the j^{th} row of the reconstruction layer to all the h' neurons of the last layer of the decoder network. In this paper, we use $[\boldsymbol{W}^{(r)\top}]_j = \tanh(\boldsymbol{W}_{\boldsymbol{\omega}_r}\boldsymbol{\phi}(\boldsymbol{u}_j))$, where $\boldsymbol{W}_{\boldsymbol{\omega}_r} \in \mathbb{R}^{h'\times b}$ is the model parameter for the tiny network.

4 Empirical Evaluation

We compared FsNet with CAE (Balin et al., 2019), which is a unsupervised, neural-network-based, feature selection method, Diet Networks (Romero et al., 2017), HSIC Lasso (Yamada et al., 2014, 2018a; Climente-González et al., 2019), and mRMR (Peng et al., 2005). Notably, CAE and HSIC Lasso are state-of-the-art, nonlinear feature selection methods, which are deep and shallow, respectively. FsNet and CAE (Balin et al., 2019) were run on a Linux server with an Intel Xeon CPU Xeon(R) CPU E5-2690 v4 @ 2.60 GHz processor, 256 GB RAM, and NVIDIA P100 graphics card. HSIC Lasso (Yamada et al., 2014) and mRMR (Peng et al., 2005) were executed on a Linux server with an Intel Xeon CPU E7-8890 v4 2.20 GHz processor and 2 TB RAM.

For FsNet and CAE, we conducted experiments on all the datasets using a fixed architecture, defined as $[d \to K \to 64 \to 32 \to 16(\to |\mathcal{Y}|) \to 32 \to 64 \to d]$, where d and $|\mathcal{Y}|$ are data dependent, and $K \in \{10, 50\}$. Each hidden layer uses the *leakyReLU* activation function and *dropout* regularization with a dropout rate of 0.2. We implemented FsNet in *keras* and used the *RMSprop* optimizer for all the experiments. For the regularization parameter λ, we used $\lambda = 1$ for all the experiments. We performed the experiments with 6000 epochs at a learning rate of $\eta = 10^{-3}$, initial temperature $\tau_0 = 10$, and end temperature $\tau_E = 0.01$ in the annealing schedule for all the experiments.

We studied a metagenome dataset (Lloyd-Price et al., 2019), which contains information regarding the gut bacteria of 359 healthy individuals and 958 patients with inflammatory bowel disease. Specifically, 7 547 features are KEGG orthology accession numbers, which represent molecular functions to which reads from the guts of samples guts are mapped. We included three additional features: age, sex, and race.

Table 1: Classification accuracy of different methods on the metagenome dataset on inflammatory bowel disease.

	Accuracy	
Method	$K = 50$	$K = 100$
FsNet	0.907 ± 0.017	0.917 ± 0.014
CAE	0.750 ± 0.065	0.700 ± 0.067
HSIC Lasso (B=10)	0.931 ± 0.006	0.931 ± 0.006
HSIC Lasso (B=20)	0.942 ± 0.002	0.943 ± 0.002
mRMR	0.944 ± 0.002	0.933 ± 0.002
SVM	0.876 ± 0.002	
Diet-networks	0.902 ± 0.023	

We selected 50 or 100 features on this dataset using FsNet, CAE, HSIC Lasso, and mRMR. For HSIC Lasso, as the number of samples was high, we employed the block HSIC Lasso (Climente-González et al., 2019), where B denotes the tuning parameter of the block HSIC Lasso, and $B = n$ is equivalent to the standard HSIC Lasso (Yamada et al., 2014). The Diet-networks outperformed SVM. The proposed FsNet could achieve better performance than Diet-networks with significantly less number of features, while CAE failed to train the model due to the large number of model parameters. Moreover, the model compression ratio between FsNet and CAE is 21.41, and thus we conclude that FsNet can obtain preferable performance with much less number of parameters for high-dimensional data.

References

Muhammed Fatih Balin, Abubakar Abid, and James Y. Zou. Concrete autoencoders: Differentiable feature selection and reconstruction. In *ICML*, 2019.

Héctor Climente-González, Chloé-Agathe Azencott, Samuel Kaski, and Makoto Yamada. Block HSIC Lasso: model-free biomarker detection for ultra-high dimensional data. *Bioinformatics*, 35 (14):i427–i435, 07 2019.

G. E. Dahl, J. W. Stokes, Li Deng, and Dong Yu. Large-scale malware classification using random projections and neural networks. In *ICASSP*, 2013.

Yixue Li and Luonan Chen. Big biological data: Challenges and opportunities. *Genomics, Proteomics & Bioinformatics*, 12(5):187–189, 2014.

Shuangli Liao, Quanxue Gao, Feiping Nie, Yang Liu, and Xiangdong Zhang. Worst-case discriminative feature selection. In *IJCAI*, 2019.

Jason Lloyd-Price, Cesar Arze, Ashwin N Ananthakrishnan, Melanie Schirmer, Julian Avila-Pacheco, Tiffany W Poon, Elizabeth Andrews, Nadim J Ajami, Kevin S Bonham, Colin J Brislawn, et al. Multi-omics of the gut microbial ecosystem in inflammatory bowel diseases. *Nature*, 569(7758): 655–662, 2019.

C. J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017.

Vivien Marx. The big challenges of big data. *Nature*, 498(7453), 2013.

Mahdokht Masaeli, Glenn Fung, and Jennifer G. Dy. From transformation-based dimensionality reduction to feature selection. In *ICML*, 2010.

Di Ming and Chris Ding. Robust flexible feature selection via exclusive L21 regularization. In *IJCAI*, 2019.

Hanchuan Peng, Fuhui Long, and Chris H. Q. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE TPAMI*, 27(8):1226–1238, 2005.

Adriana Romero, Pierre Luc Carrier, et al. Diet networks: Thin parameters for fat genomics. In *ICLR*, 2017.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal Stat. Society*, 58(1): 267–288, 1996.

Pascal Vincent, Hugo Larochelle, et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11:3371–3408, 2010.

Piotr Iwo Wójcik and Marcin Kurdziel. Training neural networks on high-dimensional data using random projection. *PAA*, 22(3):1221–31, 2019.

Makoto Yamada, Wittawat Jitkrittum, Leonid Sigal, Eric P Xing, and Masashi Sugiyama. High-dimensional feature selection by feature-wise kernelized lasso. *Neural computation*, 26(1):185–207, 2014.

Makoto Yamada, Jiliang Tang, et al. Ultra high-dimensional nonlinear feature selection for big biological data. *IEEE TKDE*, 30(7):1352–1365, 2018a.

Makoto Yamada, Yuta Umezu, Kenji Fukumizu, and Ichiro Takeuchi. Post selection inference with kernels. In *AISTATS*, 2018b.

Xiucai Ye, Hongmin Li, Akira Imakura, and Tetsuya Sakurai. Distributed collaborative feature selection based on intermediate representation. In *IJCAI*, 2019.